CICS/ ITEC 385 Fall 2014

Project , Due by Dec. 11th 2014 (Firm deadline)

This is a programming project will allow you to obtain experience working with block ciphers and encryption modes. You can write a program in a language of your choice and can use a cryptographic library of your choice.

Options for cryptographic libraries that you might useful include:
(a) Libgcrypt, available from http://www.gnu.org/software/libgcrypt/.
(b) OpenSSL library. This is already installed on many machines, so you can use man pages (e.g., man crypto or more specific commands such as man des). The documentation can be accessed online from http://www.openssl.org/, but it is incomplete and outdated (e.g., there are no manual pages for AES), so some of the functions will have to be looked up from the header _les (usually located in /usr/include/openssl).

(c) Crypto++ library, available from http://www.cryptopp.com/.
(d) Java built-in cryptographic library (Java might be a sub-optimal choice for this assignment).
(e) MIRACL library, available from http://www.certivox.com/miracl/.
For this project, you are to implement the following functionalities:
(a) AES key generation, encryption, and decryption using 128-bit key in one of the secure modes;
(b) AES key generation, encryption, and decryption using 128-bit key in CFB or OFB mode (this mode has to be different from the one used in (a)) to be used as a stream cipher with messages of size 1 byte (i.e., r = 8 in the description of CFB and OFB modes) as well as full block messages. The same functionality is also to be implemented for messages of size 1 bit (r = 1).

There are restrictions on what functions from a cryptographic library can be called. In particular, library functions that you can call are plain single-block AES encryption and decryption (i.e., encrypting a one-block message in the ECB mode) and a pseudo-random number generator. This means that you will need to implement both encryption modes yourself (one in the stream mode with the message size smaller than the block size), using function calls that implement AES block encryption and decryption algorithms.

All values that are to be chosen at random can be produced using a cryptographically strong pseudo-random number generator (which normally comes with a cryptographic library), but you'll need to properly initialize it. Include checking for correctness (e.g., that computed ciphertexts decrypt to the original data).

The programs should run (and will be evaluated). All values that are to be chosen at random can be produced using a cryptographically strong pseudo-random number generator (which normally comes with a cryptographic library), but you'll need to properly initialize it. Include checking for correctness (e.g., that computed ciphertexts decrypt to the original data).

After the algorithms are implemented, the second part of this project asks you to measure the speed of each implementation. Create two files: one of size 50 bytes and another of size 1MB (the content of the files can be arbitrary) and measure:

(a) the time to generate an AES key;

(b) the speed of encryption and decryption for mode (a) above for both files, given as the total time and also as throughput in bytes per second;

(c) the speed of encryption and decryption for mode (b) above for both files using 1-bit (60622 only), 8-bit, and full block message sizes, given as the total time and also as throughput in bytes per second.

Include the results of your measurements/computations and the source code in the homework that you turn in during the class. Also, submit the source code together with instructions on how to run your program (and install a cryptographic library if necessary) in a README file via the course dropbox. The submission needs to include a Make file that will produce an executable from the submitted source code on student machines if the language you are using requires compilation. If the program doesn't compile or run, you are risking getting zero credit.